How (Not) to Get the Vendor Driver Merged Upstream

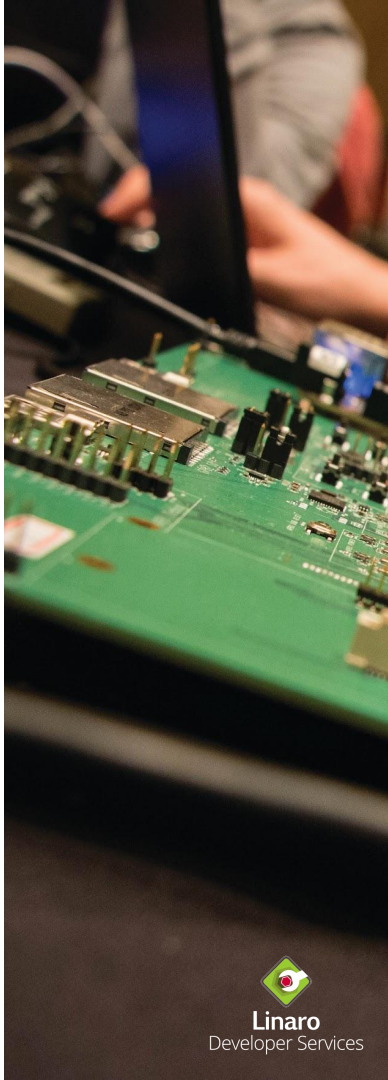
Harmful advices from the kernel developer

Dmitry Baryshkov
mailto:dmitry.baryshkov@linaro.org

Linaro
Developer Services

About me

- Linux kernel contributor since 2007
  - drm/msm co-maintainer
  - Around 2300 commits
  - Regularly hitting top-20 contributors
  - Significant time spent on vendor drivers and on reviewing submissions
- OpenEmbedded contributor since 2007
- also GNUTLS, Nettle, Rust-crypto, etc.



Linaro
Developer Services

Legend

- Bad practice and harmful advices
+ Best current practice

`Command to run`

Zero

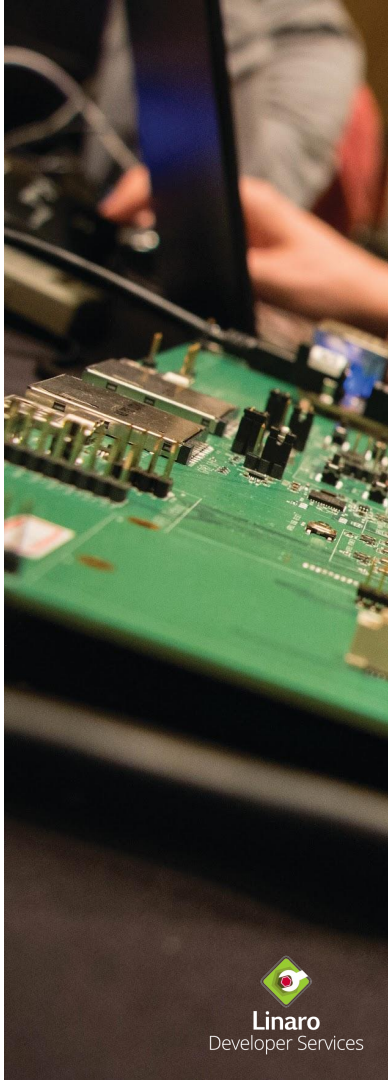
Don’t upstream it at all!

Linaro
Developer Services

Attempt Zero

We have existing driver for ImportantDevice

Let’s keep it out the tree and never send it upstream



Attempt Zero - don’t send upstream

- It’s not good enough
- It’s not important enough
- Nobody cares
- We can maintain it ourselves
- We can break some of the API and existing rules!

Attempt Zero - don’t send upstream

- It’s not good enough
- It’s not important enough
- Nobody cares
- We can maintain it ourselves
- We can break some of the API and existing rules!

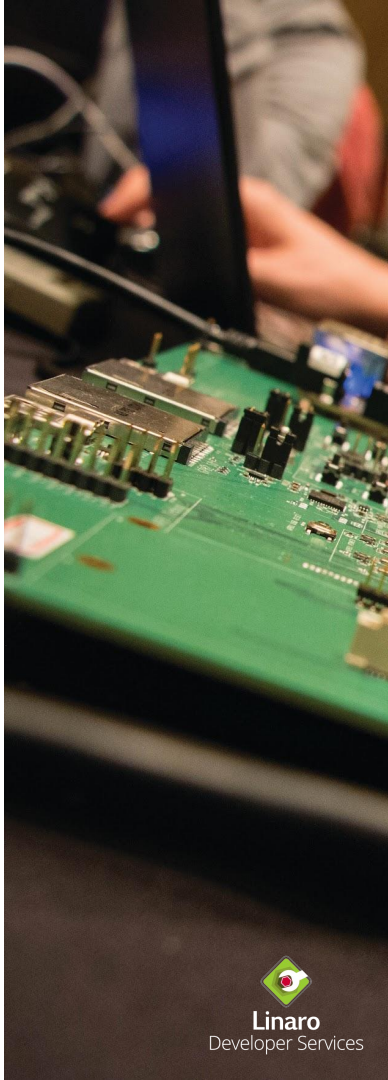
- Maintenance burden. Kernel developers are constantly changing API
- Breaking API means breaking assumptions of existing software
- Base platform code can get dropped as maintainers see no interest

- Usually the driver becomes abandonware or doesn’t satisfy users

Attempt ZeroPlus

We have existing **kernel tree** for ImportantDevice

Let’s keep it and never push it upstream



Attempt ZeroPlus - we have a tree

- We can maintain it ourselves
- We can break some of the API and existing rules!

Attempt ZeroPlus - we have a tree

- We can maintain it ourselves
- We can break some of the API and existing rules!

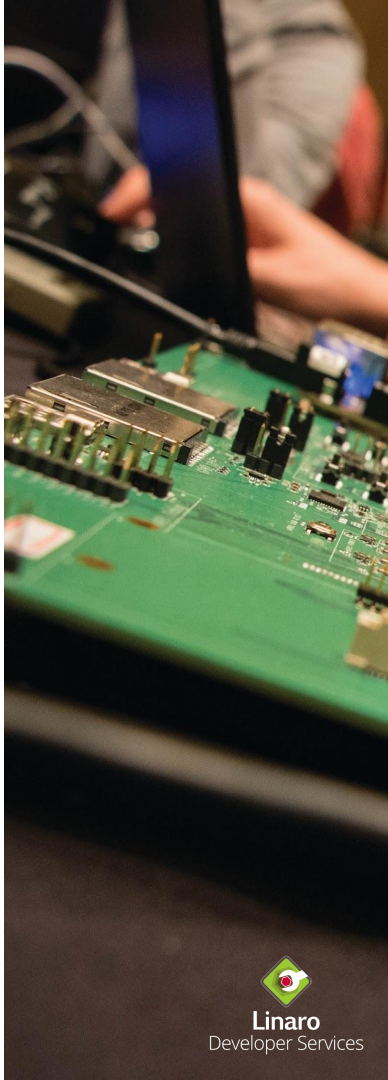
- Maintenance burden. No way to upgrade to the next revision
- Breaking API means breaking assumptions of existing software


Let’s upstream it

Attempt One

We have existing driver for ImportantDevice

It has been written by FamousVendor and it has been tested on a variety of devices. Let send it AS IS!



Attempt One - send AS IS

Can other kernel developers maintain it?

- Does it follow Linux code style?
  - CamelCase? spaghetti code? 5-6 levels of indentation? A function of 500 lines?
  + https://docs.kernel.org/process/coding-style.html
  + `` `./scripts/checkpatch.pl` ``
- Does it compile?
  - ARM, MIPS, PowerPC... S/390?
  - `` `make W=1` ``?
- Does it behave well?
  - Will it crash if there is no hardware?
  - Will it crash if there is no memory for allocation?
  - Will it crash if user sends wrong command?
  - Will it spam logs if user sends wrong command?
  - Will it ... ?

Attempt One - preparing the patches

Before sending things out, prepare the patches.

- Use b4 tool
  - https://b4.docs.kernel.org/en/latest/
- Baseline
  - ‘We have tested this driver on last year LTS kernel’
  - ‘I have used 5.15 kernel to run the tests’
  + Use either the latest Linus’s tree or linux-next
  + `b4 prep -n your-driver-name`
- Is the driver complete?
  + Document all public API that you have added to get your driver to work
  + DeviceTree platforms:
    - document DT additions (including new properties!)
    - Make sure that `make dt_bindings_check` pass.
    - And that `make dtbs_check` doesn’t bring new warnings
- Does it fit?
  - Take a look around. Check similar drivers.

Attempt One - preparing the patches

Split the code. Logically. Atomically. Sensibly.

- This driver has 100 kLoC
- It comes in 30 patches
  - Or a single patch
- Makefile is added in patch 29

- ‘This patch is required to make it compile with the changes from patch #’

+ The kernel should build and work after each and every patch in your tree
+ It should be possible to review and understand your work.
+ Descript why are you introducing the change, not what is getting done
+ Can you limit the scope of the patchset?

Attempt One - Sending it out

- I think it is ready. Can open a Merge Request?
- Or ZIP all patches and attach them to a single mail?
- Or post them on my website and ask kernel developers to take a look?

+ Linux kernel uses email-based submission system. No exceptions.
+ Single patch per email
  - https://docs.kernel.org/process/submitting-patches.html
  - https://git-scm.com/docs/user-manual
  - Some subsystems have slightly different rules
+ `b4 prep --auto-to-cc --edit-cover`

+ `b4 send`

+ Wait for the feedback
+ You patches will be reviewed. And here comes a tricky part

Attempt One - Feedback

- Ignore maintainer comments
- Skip maintainer questions
- If you don’t understand the question, just ignore it
- Spam them. Send new iteration for every small change.

Attempt One - Path to Attempt Two

+ Pick up Reviewed-By tags
  + `b4 trailers -u`
+ It is fine to disagree with the reviewers. Respond, question, propose, etc.
+ Let the discussion come to conclusion
+ Implement the changes
  + `git commit && b4 prep --edit-cover`

+ Give everybody a chance to review and respond.
  + No more than 1 iteration in 24 hours
  + For a big patchset, give it a week

+ `b4 send`

Attempt One - Being ignored

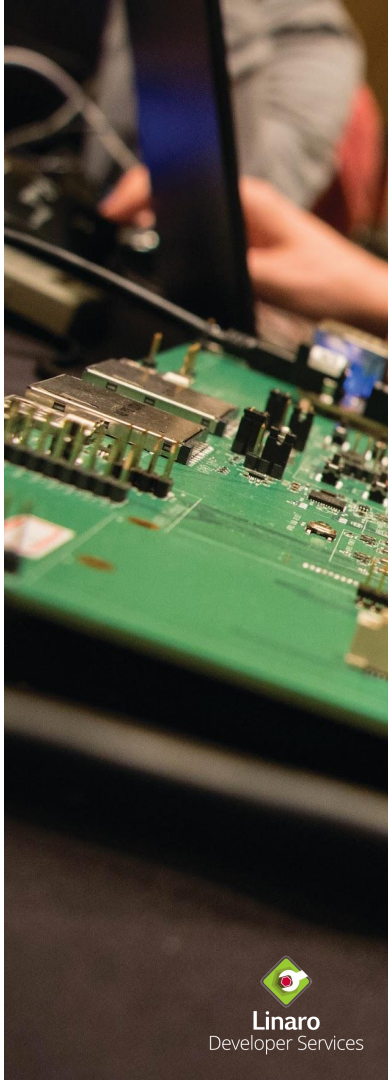
No response at all. Is it useless?

+ Sometimes it takes more time to review
+ Check recipients list
+ After two weeks you can do `b4 send --resend`.
+ Or ping a maintainer.

Summary

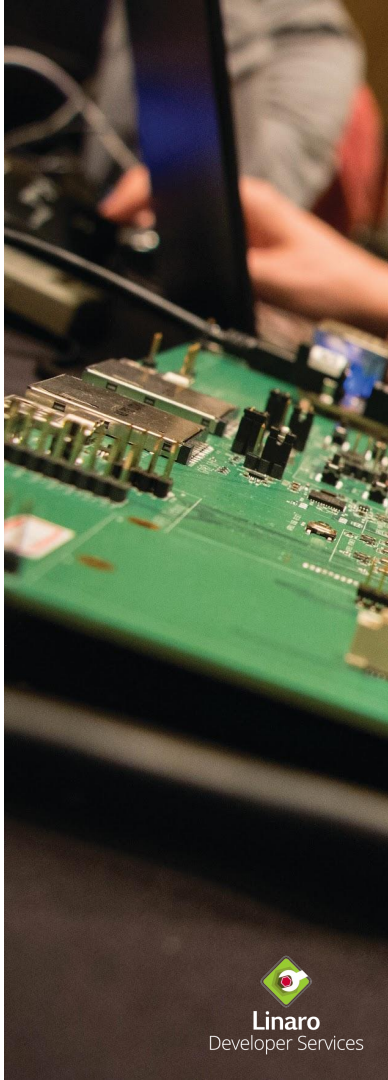
- Getting the driver in is an iterative process
  - ‘v10’ are not frequent, but happen from time to time
- Be prepared to significantly rewrite original driver
  - sometimes complete rewrite is required
- Be motivated to do the work
  - getting the code in pays off in the long run
- Forget about the deadlines and estimates
  - one can not predict the number of review cycles

- Better to ask a question rather than to show ignorance



Links

- Kernel Documentation
  - https://docs.kernel.org/process/development-process.html
  - https://docs.kernel.org/process/coding-style.html
  - https://docs.kernel.org/process/submitting-patches.html
- Git manual
  - https://git-scm.com/doc
- Git trees
  - https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
  - https://git.kernel.org/pub/scm/linux/kernel/git/next/linux-next.git



Thank you

Questions?
Failure stories?
Success stories!

Linaro
Developer Services